Pauli von Boehm

Tietokantasovelluksen kehittäminen

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Kone- ja tuotantotekniikka

Insinöörityö

10.5.2012

Metropolia

Tekijä Otsikko Sivumäärä Aika	Pauli Tuomas von Boehm Tietokantaohjelmiston kehittäminen 28 sivua 25.4.2013
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Kone- ja tuotantotekniikka
Suuntautumisvaihtoehto	Koneautomaatio
Ohjaaja	Lehtori Jari Savolainen

Tämä insinöörityö suoritettiin toimeksiantona Veranos Oy:lle ja se on jatkoa aiemmin aloitetulle kehitysprojektille. Tarkoituksena oli kehittää sovellus, jonka avulla hydraulisille pumpuille tehtyjä mittauksia pystytään havainnoimaan reaaliaikaisesti näytöltä tilavuusvirran ja paineen suhteena. Sovellus on yhteydessä tietokantaan, johon mittaustulokset ja pumppujen tiedot tallennetaan ja josta tietoja voi myös hakea myöhempää tarkastelua varten. Varsinaisten mittausjärjestelyiden toteuttamiseen ei keskitytty, mutta sovellukseen luotiin mahdollisuus simuloida niitä.

Työssä perehdytään Sql-kielisten relaatiotietokantojen toimintaan ja rakenteeseen sekä siihwn, miten tietokannan saa liitettyä osaksi Qt Creatorilla ohjelmoitua sovellusta. Oleellista oli selvittää, miten ja millä komennoilla tietokannassa olevia tietoja voi muokata ja mitä sääntöjä siinä on otettava huomioon. Itse sovelluksen kehittämisessä keskityttiin mittausten visuaaliseen esitystapaan ja tietokannassa olevien tietojen tehokkaaseen käsittelyyn. Mittausten esittäminen tapahtuu piirtämällä mittaustulokset käyränä koordinaatistoon, mitä varten Qt Creatoriin oli asennettava piirtämisen mahdollistava työkalu. Sovelluksen ja tietokannan yhdistämisessä pyrittiin luomaan käyttäjälle mahdollisimman selkeä toimintaympäristö, jossa tietojen käsittely on loogista ja helppoa.

Työ pyrittiin dokumentoimaan mahdollisimman kattavasti etenkin sovelluksen koodin osalta. Dokumentointiin panostettiin, jotta henkilön, joka toteuttaa lopulliset mittausjärjestelyt, olisi helppo liittää ne osaksi tämän työn yhteydessä luotua ohjelmaa.

Avainsanat	Sql, relaatiotietokanta, Qt Creator

Author Title Number of Pages Date	Pauli Tuomas von Boehm Development of Database Application 28 pages 25 April 2012
Degree	Bachelor of Engineering
Degree Programme	Mechanical Engineering
Specialisation option	Machine Automation
Instructor	Jari Savolainen, D.Sc (Tech.)

This Bachelor's thesis was carried out as an assignment for Veranos Oy, and it is a continuation of previously begun development project. The aim was to develop an application that allows observing the measurements of the volumetric flow and pressure ratio, made for hydraulic pumps. The application is connected to a database, in which the measurements and information of the pumps are stored and from which data can also be retrieved for analysis. This study does not, however, focus on the implementation of actual measurement arrangements but the application has an option to simulate them.

This thesis studies the operation and structure of relational databases written in sql language and how to connect the database to the application programmed with Qt Creator. It was essential to find out how the commands in the database can be edited and what rules must be taken into account. The development of the application focused on the visual presentation of the measurements and efficient processing of the information in the database. Combining the application and the database aimed to create a clear framework for a user in which handling of the data is logical and easy.

Furthermore, the objective was to document the project work as comprehensively as possible, especially the application code. It is important to focus on documenting, so it would be easier for the person who carries out the final measurement arrangements, to include his/her work with the application created in this work.

Keywords	Qt Creator, database, hydraulic pumps

Sisällys

Lyhenteet

SQL	Tietokantakieli
C++	Ohjelmistokieli
Qt	C++ -kehitysympäristö

1 Johdanto

Tämä insinöörityö tehtiin Veranos Oy:lle tarkoituksena kehittää heidän mahdollisuuksiaan tarjota laadukkaita hydraulisten pumppujen huoltopalveluita. Veranos Oy perustettiin vuonna 1997, jolloin toiminta keskittyi lähinnä varaosamyyntiin. Yrityksen aputoiminimenä on Varaosaparatiisi. Vuosien saatossa yrityksen toiminta on kehittynyt ja nykyään yritys keskittyy pääasiassa kaivinkonehydrauliikkaan. Toimintaan tuli mukaan ensin kaivinkoneiden vetonapojen huolto, ja tietämyksen lisäännyttyä, myös erilaiset hydraulistoimiset pumput ja moottorit. Veljekset Rauno ja Pertti Martikainen ovat yhtiön pääomistajat. Tällä hetkellä yritys työllistää neljä henkeä. Yrityksen liikevaihto on vuosittain noin miljoona euroa.

Laadukas hydraulisten laitteiden huolto on toiminnan kulmakivi. Veranos Oy:n huoltotoiminta perustuu nykyään avoimeen piiriin perustuvien hydraulisjärjestelmien huoltoon. Yrityksen tavoitteena on laajentaa toimintaansa myös suljetun piirin järjestelmiin. Markkinoilla on tarvetta suljetun piirin hydraulisjärjestelmien asiantuntijoille, joten askel tähän suuntaan laajentaisi yrityksen asiakaspiiriä huomattavasti. Tämän työn lopputuloksena on tarkoitus luoda tietokantasovellus, johon Veranos Oy voi kerätä tietoja huoltamistaan laitteista ja niille tehdyistä testeistä sekä testien tuloksista. Tämä vaatii perehtymistä tietokantarakenteisiin ja niiden toimintaan.

Sovellukseen sisällytetään myös mahdollisuus simuloida hydraulisille pumpuille suoritettavaa mittaustapahtumaa, vaikka varsinaisia mittausjärjestelyjä ei tämän työn yhteydessä toteuteta. Suoritettavat mittaukset kuvastavat pumppujen mukana toimitettuja esikäyriä, jotka esittävät pumpun toimintaa eri olosuhteissa. Sovelluksen toteutusta varten on tutkittava mahdollisuuksia esittää visuaalisesti näytöllä mittaustuloksia, sekä esikäyriä pumpun mallista tai testattavasta ominaisuudesta riippumatta.

2 Tietokannat

Tietokanta on moniselitteinen käsite. Yleisesti ottaen tietokanta on loogisesti yhteenkuuluvien, tallennettujen tietojen joukko, jota voidaan helposti käsitellä tietokantakielellä. Nykyään käytetään hyvin yleisesti SQL-pohjaisia relaatiotietokantoja, joiden tarkasteluun keskitytään myös tässä työssä. (1, s. 4.)

2.1 Tietokannan hallintajärjestelmä

Tietokannassa olevia tietoja hallinnoi tietokannan hallintajärjestelmä (Data Base Management System DBMS). Hallintajärjestelmien avulla voidaan kehittää hyvin laajoja kokonaisuuksia, joiden avulla voidaan käsitellä reaaliaikaisesti suurta määrää tietoa monesta eri päätteestä käsin. Yleisesti käytettyjä hallintajärjestelmiä ovat mm. MySql, Sqlite, Oracle ja PostgreSql. Tämän työn yhteydessä tehdyn ohjelmiston toteutukseen käytettiin PostgreSql-hallintajärjestelmää.(1, s. 4.)

Tietokannan hallintajärjestelmän voi liittää myös osaksi eri ohjelmointikielillä toteutettuja ohjelmistoja. Tällöin ohjelmistosta on avattava yhteys tietokantaan ja käsiteltävä siellä olevia tietoja SQL-kielisillä komennoilla. (1, s. 10.)

Tässä työssä PostgreSQL liitettiin osaksi C++ -ohjelmointikielellä toteutettua ohjelmistoa. Tämä toteutettiin asentamalla PostgreSQL:n käytön mahdollistavat liitännäiset (plugin) ohjelman kehittämiseen käytettyyn Qt-kehitysympäristöön .

2.2 PostgreSQL

PostgreSQL on hyvin kehittynyt avoimen lähdekoodin tietokannan hallintajärjestelmä, jota on kehitetty vuosien aikana eri nimisinä projekteina 1970-luvun lopusta lähtien. Kehitystyö on tapahtunut pääasiassa Kalifornian yliopistossa Berkeleyssa, jolle ohjelmiston lisenssi myös kuuluu. Avoimen lähdekoodin myötä ohjelma on ilmainen ja vapaasti muokattavissa.(2, s. 7 - 10.)

PostgreSQL on oliorelaationaalinen tietokantajärjestelmä, mikä tarkoittaa, että tavallisesta relaatiotietokannasta poiketen sillä on myös olio-ohjelmoinnista tuttuja piirteitä, esimerkiksi taulujen välisiä vektoreita ja osoittimia. Tässä työssä näitä ominaisuuksia ei kuitenkaan käytetty, joten tietokantoja keskitytään käsittelemään tavallisina relaatiotietokantoina.(2, s. 7 - 10; 1, s 6)

2.3 Relaatiotietokannat

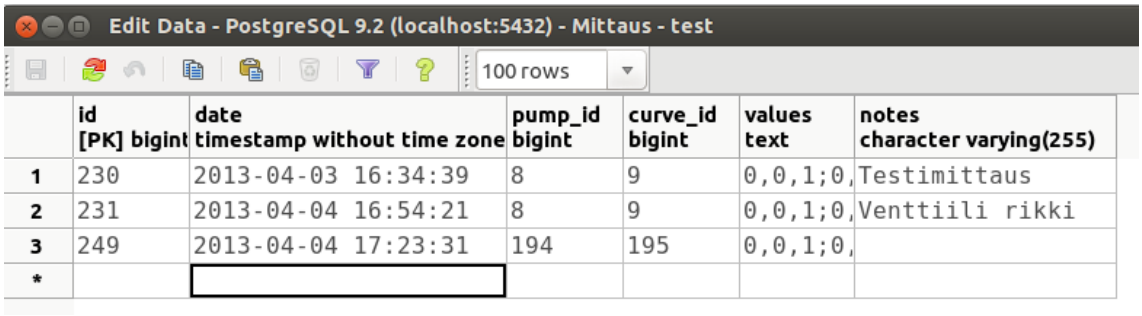
IBM:n tutkija E.F. Coddin julkaisi vuonna 1970 relaatiomallin, johon nykyiset relaatiotietokannat perustuvat. Relaatiomalli on tietomalli, jonka avulla voidaan kuvata tietorakenteita ja niiden välisiä yhteyksiä helposti ja yksinkertaisesti, mikä on johtanut SQL-kielen suureen suosioon. SQL onkin lähes syrjäyttänyt aiemmin käytetyt hierarkkiset ja verkkomalliset tietokantatyypit. Siitä on myös laadittu kansainvälisiä standardeja kuten ANSI SQL-92 ja SQL:2003. (2, s 40; 1, s 10.)

Relaatiomallin perustana on joukko-oppi. Sen avulla suureen määrään tietoja voidaan kohdistaa joukko-operaatioita, jotka muokkaavat tietoa tehokkaasti. Relaatiomalli ei määritä tietokannan fyysistä tuoteutustapaa. Sen hoitaminen on jätetty tietokannan hallintajärjestelmien toimittajille. (1, s 7-8.)

2.4 Relaatiotietokannan rakenne

Relaatiotietokannan perusrakenne koostuu yhdestä tai useammasta taulusta (table), johon voi tallentaa käsiteltävää tietoa. Yksi taulu käsittää yleensä yhden tietokokonaisuuden. Taulut on jaettu riveihin (rows) ja sarakkeisiin (columns), joiden avulla tietoa jäsennetään. Sarakkeet jakavat rivien tiedon komponentteihin, joilla on määrätty nimi ja tietotyyppi (datatype). Tietotyypit määrittävät millaista tietoa kuhunkin sarakkeeseen voi tallentaa. (2, s. 40.)

Kuvassa 1 on esitetty tämän työn toteutukseen käytetty test-taulu, johon on tallennettu tiedot simuloiduista mittauskäyristä. Jokaisella rivillä on yhden mittauksen tiedot. Sarakkeilla on määritelty, mitä tietoa yhden mittauksen yhteydessä tallennetaan ja missä muodossa. Esimerkiksi values-sarakkeen tietotyyppi on text, mikä tarkoittaa, että kyseiseen sarakkeeseen voi tallentaa tekstiä ja numeroita.



	id [PK] bigint	date timestamp without time zone	pump_id bigint	curve_id bigint	values text	notes character varying(255)
1	230	2013-04-03 16:34:39	8	9	0,0,1;0,	Testimittaus
2	231	2013-04-04 16:54:21	8	9	0,0,1;0,	Venttiili rikki
3	249	2013-04-04 17:23:31	194	195	0,0,1;0,	
*						

Kuva 1. Test-taulu

2.5 Tietotyypit

Jokaisella sarakkeella on aina oma tietotyyppinsä. Tietotyyppejä on paljon erilaisia, kuten esimerkiksi tekstiä ja lukuja (text), kokonaislukuja (int) ja totuusarvoja (bool) sisältäviä. Niiden avulla tiedon jäsennys helpottuu ja jokaisessa sarakkeessa on varmasti tallennettuna oikean tyyppistä tietoa.

Tietotyyppi määrittää myös, kuinka suuren määrän tilaa kyseinen tallennuspaikka voi maksimissaan käyttää. Esimerkiksi test-taulun (kuva 1) notes-sarakkeen “character varying (255)” -tietotyyppi voi tallentaa maksimissaan 255 merkkiä. Tiedon tallennus kuitenkin tehostuu jos tiedon tallentamiseen varataan tilaa vain käytettyjen merkkien verran. (2, s. 41 – 43.)

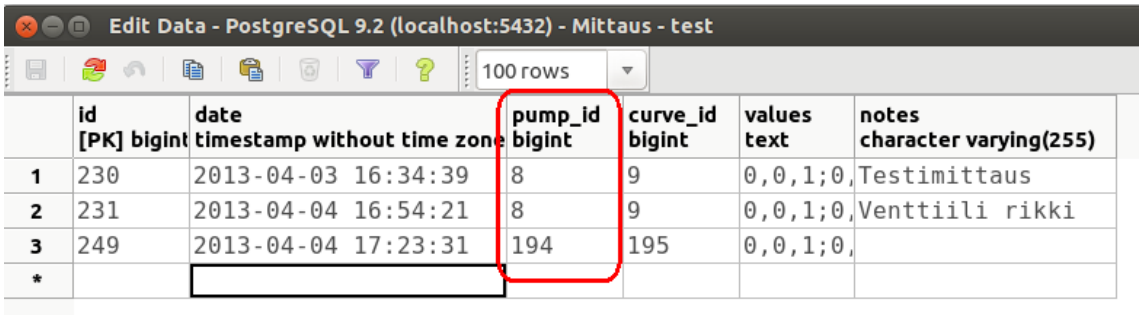
2.6 Perus- ja viiteavain

Relaatiotietokannan jokaisella taululla on oltava perusavain (primary key). Perusavain varmistaa, että jokaisella rivillä on uniikkia tietoa, mikä takaa tietokannan tiedon eheyden. Kun jokainen tallennettava tieto (rivi) on yksilöity, tarvittava tieto löytyy aina oikeasta paikasta, jolloin sen käsittely on tehostuu. (2, s 40.)

Kuvassa 2 perusavain on ensimmäinen sarake nimeltään id. Perusavaimen tunnus näkyy tietotyypin yhteydessä [PK]-merkintänä. Kuten on havaittavissa, jokaisella rivillä on yksilöllinen id, joten jokainen mittaus on tallennettu yksilöllisesti.

Kun yhdessä taulussa oleva tieto on yhteydessä toisessa taulussa olevaan tietoon, käytetään apuna viiteavainta (foreign key). Viiteavain varmistaa, että kahteen tai useampaan tauluun tallennetut tiedot ovat keskenään järjestyksessä. Viiteavainten käyttö on erittäin hyödyllistä kun rakennetaan suuria ja monimutkaisia tietokantoja, joissa useat taulut ovat linkittyneet toisiinsa. (2, s 41)

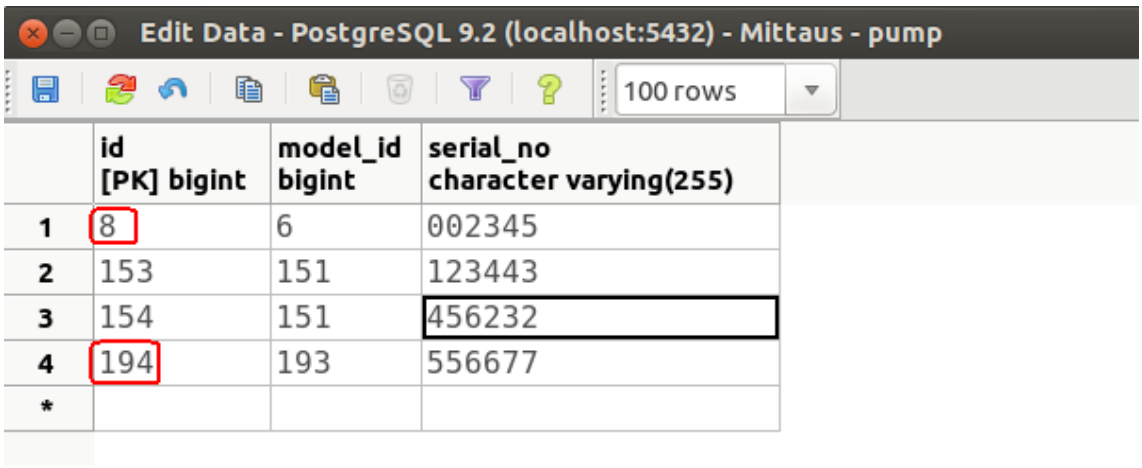
Kuvan 2 test-taululla on viiteavain pump_id sarakkeessa. Sen tunnusta ei näytetä kuten perusavaimen, mutta tieto viiteavaimen käytöstä on määritelty sarakkeen tiedoissa. Tässä tapauksessa pump_id:n viiteavain on yhdistetty pump-taulun (kuva 3) id-sarakkeeseen.



	id [PK] bigint	date timestamp without time zone	pump_id bigint	curve_id bigint	values text	notes character varying(255)
1	230	2013-04-03 16:34:39	8	9	0,0,1;0,	Testimittaus
2	231	2013-04-04 16:54:21	8	9	0,0,1;0,	Venttiili rikki
3	249	2013-04-04 17:23:31	194	195	0,0,1;0,	
*						

Kuva 2. Test-taulun viiteavain

Pump_id-sarakkeen viiteavain varmistaa, että tallennettu mittaus on varmasti tehty pumpulla, joka on jo tallennettu tietokantaan. Tämä tarkoittaa sitä, että pump_id-sarakkeen arvon on vastattava pump-taulussa olevaa id-sarakkeen arvoa, joka on pump-taulun perusavain eli uniikki tieto. Näin mittaustuloksesta voi suoraan nähdä, millä pumpulla mittaus on suoritettu tai toisaalta mitä mittauksia tietyllä pumpulla on suoritettu.



Kuva 3. Pump-taulu

2.7 Eheytyssääntö

Relaatiomalli määrittää, miten rakennetaan eheä tietokanta. Tietokannan eheys (integrity) tarkoittaa sitä, että kaikki tietokantaan tallennettu tieto on oikeaa ja keskenään ristiriidatonta sekä vastaa realimaailmaa. Tärkein eheyssääntö on avaineheys. Se tarkoittaa, että perusavaimen arvo ei voi olla tyhjä eli NULL-arvo. (1, s. 11.)

Edellisessä esimerkissä voi havaita avaineheyden toiminnan. Pump_id- ja curve_id-sarakkeiden viiteavaimet viittaavat pump-taulun ja curve-taulun perusavaimiin, joilla on oltava yksilöllinen arvo. Näin ieto on varmasti eheää, koska sarakkeet eivät voi saada tyhjää arvoa.

Toinen tärkeä eheyssääntö on viite-eheys. Sen avulla pyritään välttämään, ettei tietokannasta löydy tarpeetonta tietoa, johon ei ole viiteyhteyttä ja joka voi myös mahdollisesti myöhemmin hankaloittaa uuden tiedon muokkaamista. Viite-eheyssääntöjä on neljä kappaletta ja ne toimivat seuraavasti:

- R = Restrict eli estosääntö. Taulussa olevaa riviä ei voi poistaa, jos jokin siinä esiintyvä sarake on jonkin toisen taulun viiteavain.
- C = Cascade eli vyörytyssääntö. Jos taulusta poistetaan rivi, jonka jokin sarake toimii viiteavaimena toiseen tauluun, poistetaan myös toisesta taulusta rivit, joihin viiteavain viittaa.
- N = Set NULL eli tyhjäyssääntö. Jos taulusta poistetaan rivi, jonka jokin sarake toimii viiteavaimena, asetetaan rivien, joihin viiteavain viittaa, arvoksi tyhjä eli NULL.
- D = Set Default eli oletusarvosääntö. Jos taulusta poistetaan rivi, jonka jokin sarake toimii viiteavaimena, asetetaan rivien, joihin viiteavain viittaa, arvoksi jokin ennalta määrätty oletusarvo.

(1, s. 22, 49 – 50.)

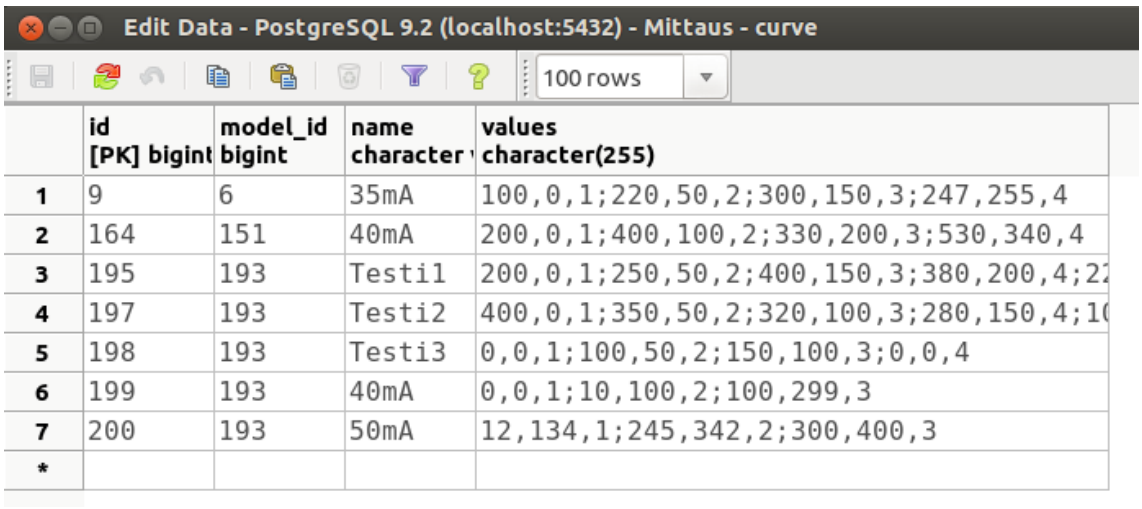
3 Mittaus-tietokanta

Mittaus-tietokanta koostuu viidestä taulusta: curve, make, model, pump ja test. Jokaisen taulun perusavaimena toimii id-sarake, joka yksilöi kaikki tallennetut rivit. Se mahdollistaa tehokkaan viiteavaimien käytön, joilla osa tauluista on linkittynyt toisiinsa.

3.1 Curve-taulu

Curve-tauluun (kuva 4) tallennetaan tiedot esikäyristä, eli pumppujen valmistajan toimittamista suorituskäyristä eri tilanteissa, sekä esikäyrän nimi ja viittaus model-tauluun. Id- sarake on taulun perusavain. Model_id-sarake on viiteavain, joka viittaa model-taulun perusavaimeen, jotta tiedetään mille pumppumallille kyseinen esikäyrä kuuluu. Name-sarakkeeseen tallennetaan esikäyrän nimi ja values-sarakkeeseen esikäyrän arvot kaksiulotteisessa koordinaatistossa.

Esikäyrien arvot on tallennettu siten, että yhteen tarkastelupisteeseen tarvitaan kolme pilkulla erotettua arvoa: x-akselin arvo, y-akselin arvo ja indeksi eli tieto kyseisestä tarkastelupisteestä. Nämä kolmen arvon joukot on erotettu puolipisteellä. Ohjelma, joka lukee tietokantaa, osaa näiden merkkien perusteella jakaa values-sarakkeen arvot osiin ja piirtää kyseisen esikäyrän näytölle.



	id [PK] bigint	model_id bigint	name character	values character(255)
1	9	6	35mA	100,0,1;220,50,2;300,150,3;247,255,4
2	164	151	40mA	200,0,1;400,100,2;330,200,3;530,340,4
3	195	193	Testi1	200,0,1;250,50,2;400,150,3;380,200,4;2
4	197	193	Testi2	400,0,1;350,50,2;320,100,3;280,150,4;1
5	198	193	Testi3	0,0,1;100,50,2;150,100,3;0,0,4
6	199	193	40mA	0,0,1;10,100,2;100,299,3
7	200	193	50mA	12,134,1;245,342,2;300,400,3
*				

Kuva 4. Curve-taulu

3.2 Make-taulu

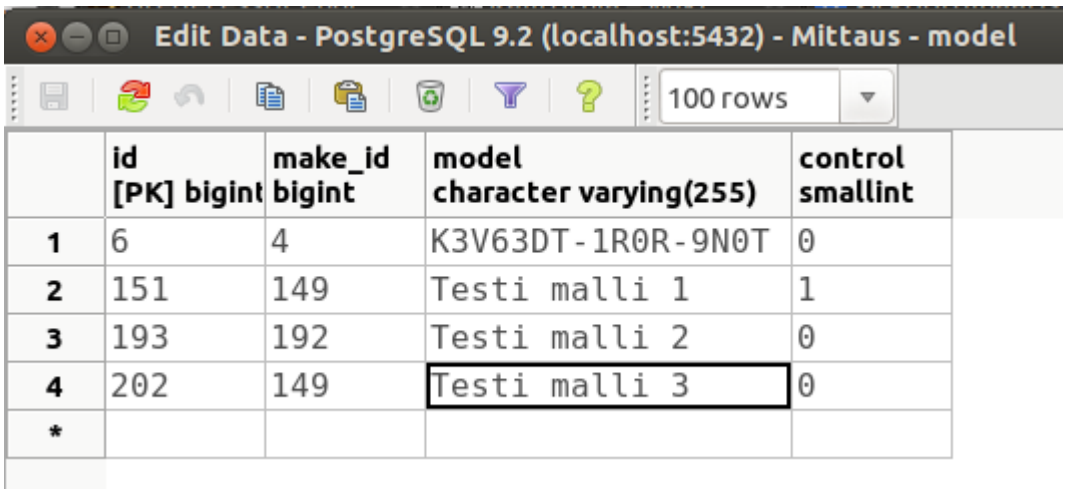
Make-tauluun (kuva 5) tallennetaan tiedot pumppujen valmistajista. Taulussa ei ole viiteavaimia mutta model-taulun make_id-sarake viittaa tämän taulun perusavaimeen eli id-sarakkeeseen.

	id [PK] bigint	name character varying(255)
1	4	KAWASAKI
2	149	REXROOTH
3	192	Valmistaja 1
4	250	Valmistaja 2
*		

Kuva 5. Make-taulu

3.3 Model-taulu

Model-tauluun (kuva 6) tallennetaan tiedot pumppujen malleista sekä siitä minkä tyyppisellä paineanturilla pumppua testataan. Id-sarake on perusavain ja make_id-sarake on viiteavain, joka viittaa make-taulun id-sarakkeeseen. Model-sarakkeeseen tallennetaan mallin nimi ja control-sarakkeeseen tieto mallin käyttämän paineanturin tyypistä. Sarakkeen arvosta ei suoraan käy ilmi, mitä paineanturia mikäkin malli käyttää, mutta tietokantaa lukeva ohjelmisto osaa tulkita kyseisen arvon oikein.



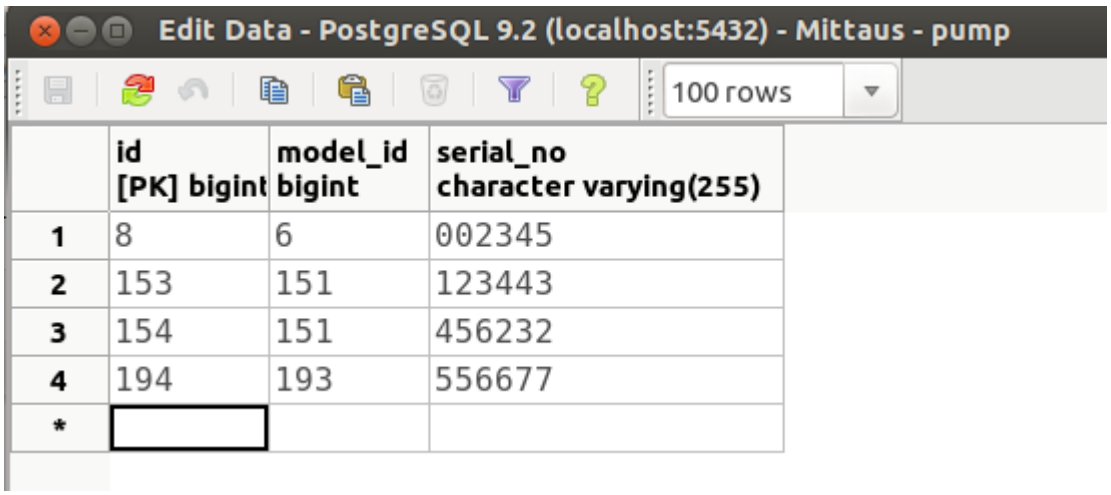
	id [PK] bigint	make_id bigint	model character varying(255)	control smallint
1	6	4	K3V63DT-1R0R-9N0T	0
2	151	149	Testi malli 1	1
3	193	192	Testi malli 2	0
4	202	149	Testi malli 3	0
*				

Kuva 6. Model-taulu

3.4 Pump-taulu

Pump-tauluun (Kuva 7) tallennetaan tieto mitattavan pumpun sarjanumerosta. Sarjanumeron perusteella voidaan pitää kirjaa juuri kyseiselle pumpulle tehdyistä mittauksista jos kyseinen pumppu tuodaan huoltoon useita kertoja tai jos se esimerkiksi vaihtaa omistajaa.

Id-sarake on taulun perusavain ja Model_id-sarake on viiteavain, joka viittaa model-taulun perusavaimeen, jotta tiedetään mistä pumppumallista on kyse kun tarkastellaan pumpun sarjanumeroa. Serial_no-sarakkeeseen tallennetaan itse sarjanumero numeroina, tekstinä tai niiden yhdistelmänä pumpusta riippuen.



	id [PK] bigint	model_id bigint	serial_no character varying(255)
1	8	6	002345
2	153	151	123443
3	154	151	456232
4	194	193	556677
*			

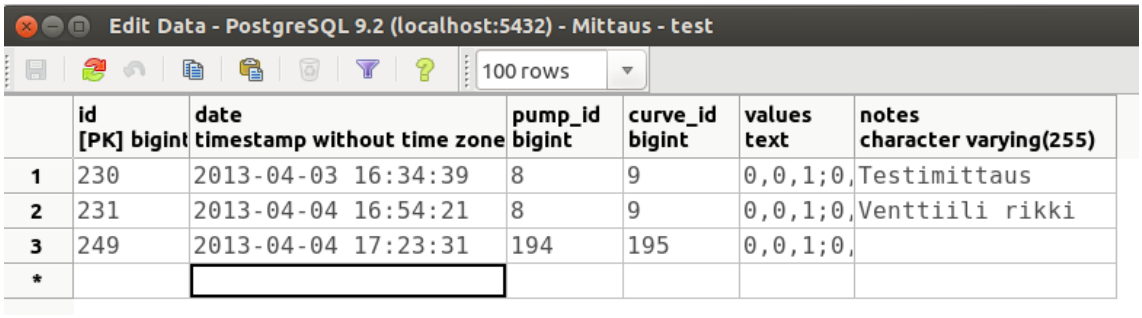
Kuva 7. Pump-taulu

3.5 Test-taulu

Test-tauluun (kuva 8) tallennetaan tiedot itse mittauksesta. Yksi mittaustapahtuma sisältää tiedot mittauksen ajankohdasta, mitatuista arvoista sekä mahdollisista mittaukseen liittyvistä muistiinpanoista.

Id-sarake on taulun perusavain. Viiteavaimia ovat pump_id sekä curve_id. Pump_id viittaa pump-taulun perusavaimeen ja määrittää näin sarjanumeron perusteella, mille pumpulle mittaus on suoritettu. Curve_id viittaa curve-taulun perusavaimeen ja kuvaa, mitä esikäyrää mittaus esittää.

Date-sarakkeeseen tallennetaan tieto mittauksen päivämäärästä ja kellonajasta. Values-sarakkeeseen tallennetaan mittauksen arvot samalla tavalla kuin esikäyrien arvot curve-tauluun. Eli yhteen tarkastelupisteeseen tarvitaan kolme pilkulla erotettua arvoa: x-akselin arvo, y-akselin arvo ja indeksi eli tieto kyseisestä tarkastelupisteestä. Nämä kolmen arvon joukot erotetaan puolipisteellä ja näitä joukkoja voi olla kymmeniä tuhansia. Ohjelma, joka lukee tietokantaa, osaa näiden merkkien perusteella jakaa values-sarakkeen arvot osiin ja piirtää kyseisen mittauksen käyrän näytölle.



	id [PK] bigint	date timestamp without time zone	pump_id bigint	curve_id bigint	values text	notes character varying(255)
1	230	2013-04-03 16:34:39	8	9	0,0,1;0,	Testimittaus
2	231	2013-04-04 16:54:21	8	9	0,0,1;0,	Venttiili rikki
3	249	2013-04-04 17:23:31	194	195	0,0,1;0,	
*						

Kuva 8. Test-taulu

4 Tietojen käsittely mittaus-tietokannassa

Jotta tietokantaan voidaan kirjoittaa tietoa tai jotta sieltä voidaan lukea tietoa, on käytettävä SQL-kielisiä komentoja (1, s 10). Mittaus-tietokantaa käsiteltäessä käytetään vain muutamaa komentoa, jotka suoritetaan erillisessä Mittaus-sovelluksessa.

4.1 Tietojen kirjoittaminen

Tietojen kirjoittaminen Mittaus-tietokantaan tapahtuu esimerkkikoodin 1 komennolla.

```
INSERT INTO table_name (column1, column2...)
VALUES (value1, value2...)
```

Esimerkkikoodi 1

Komento kertoo tietokannalle, mihin tauluun ja mihin sarakkeisiin halutut tiedot kirjoitetaan. “`INSERT INTO`” on käsky kirjoittaa tietokantaan. “`table_name`” on sen taulun nimi, jota käsitellään ja suluissa olevat `column1` ja `column2` kyseisen taulun sarakkeet, joihin tiedot kirjoitetaan. “`VALUES (value1, value2...)`” määrittää kirjoitettavat arvot siten, että `value1` kirjoitetaan `column1`:een, `value2` `column2`:een ja niin edelleen. (3.)

Esimerkkikoodin 2 komento lisää make-tauluun (kuva 5) uuden valmistajan nimeltä BOCH. id-sarake saa arvon default, koska se on taulun perusavain, jolloin se ottaa arvokseen seuraavan vapaan numeron.

```
INSERT INTO make (id, name)
VALUES (default, BOCH)
```

Esimerkkikoodi 2

4.2 Tietojen lukeminen

Tietojen lukeminen Mittaus-tietokannasta tapahtuu esimerkkikoodin 3 komennoilla.

```
SELECT column_name
FROM table_name

SELECT column_name
FROM table_name
WHERE column_name2 = value1
```

Esimerkkikoodi 3

"`SELECT column_name`" kertoo, mistä sarakkeesta tietoa haetaan siten, että "`column_name`" on sen sarakkeen (tai sarakkeiden jos haettavia kohteita on useita) nimi, josta tieto luetaan. "`FROM table_name`" kertoo, missä taulussa kyseinen sarake sijaitsee. Tämä komento hakee valitun sarakkeen kaikki arvot. (3.)

Jos hakuehtoja halutaan täsmentää, voidaan lisätä komento "`WHERE column_name2 = value1`", jolloin voidaan määrittää, että tieto haetaan vain valitun taulun sellaiselta riviltä, jossa jollain sarakkeella (`column_name2`) on haluttu arvo "`value1`". Jos määrittäviä sarakkeita on useita ne erotetaa "`AND`" sanalla. (4.)

Esimerkkikoodin 4 komento hakee test-taulun (kuva 8) date-sarakkeesta päivämäärän “2013-04-04 17:23:31”

```
SELECT date
FROM test
WHERE pump_id = 194 AND curve_id = 195
```

Esimerkkikoodi 4

5 Ohjelmointi

Veranos Oy:lle kehitettiin käyttöliittymäpohjainen sovellus, joka nimettiin Mittaus-sovellukseksi. Sen avulla voi helposti hallinnoida Mittaus-tietokannassa olevia pumppujen tietoja, sekä esittää reaaliaikaisesti pumpuille tehtyjen testien mittauskäyriä koordinaatistossa. Mittauskäyriä voi myös tallentaa tietokantaan ja hakea niitä sieltä myöhempää tarkastelua varten. Varsinaisia mittausjärjestelyjä ei tämän työn puitteissa toteutettu, vaan ne korvattiin mahdollisuutena simuloida mittaustapahtumaa. Mittausten järjestäminen toteutetaan myöhemmin erillisenä projektina ja liitetään osaksi nykyistä sovellusta.

Mittaus-sovellus kehitettiin C++ -ohjelmointikielellä QT4-kehitysympäristössä. Ohjelmointityö suoritettiin Ubuntu 12.10 -käyttöjärjestelmällä. Ubuntu valittiin sen avoimen lähdekoodin vuoksi ja koska sen Synaptic-paketinhallintaohjelmalla siihen oli helposti ladattavissa kaikki tarvittavat liitännäiset. Oleellisimpia niistä olivat Qwt-kirjasto ja PostgreSql ajurit QT4:ään, sekä Doxygen ohjelma, jolla sovelluksen koodi dokumentoitiin.

5.1 Ubuntu 12.10

Ubuntu 12.10 on tuorein versio Linux-pohjaisessa Ubuntu-käyttöjärjestelmäsarjassa. Se on avoimen lähdekoodin vapaa ohjelmisto, mikä tarkoittaa, että se on ilmainen ja sen lähdekoodi on vapaasti käytettävissä. Ensimmäinen versio Ubuntusta julkaistiin vuonna 2004, ja se on tällä hetkellä suosituin Linux-pohjainen käyttöjärjestelmä. Vaikka Ubuntu on vapaa ohjelmisto, on sille myös saatavilla ohjelmia, jotka eivät perustu vapaaseen lähdekoodiin. (5.)

5.2 Synaptic

Synaptic on Linux-käyttöjärjestelmissä toimiva graafinen paketinhallintaohjelmisto. Se suorittaa graafisen käyttöliittymän kautta sen, minkä ”apt-get” -komento suorittaa Linuxin komentorivillä eli ohjelmien ja niiden liitännäisten lataamisen ja asentamisen. (6.)

Synapticin avulla sai ladattua Mittaus-sovelluksen toiminnan kannalta tärkeät ajurit ja liitännäiset huomattavasti helpommin kuin Windows-käyttöjärjestelmästä. Tämä nopeutti huomattavasti ohjelmoinnissa tarvittavien liitännäisten toimintakuntoon saattamista.

5.3 C++

C++ on tällä hetkellä käytetyimpiä olio-ohjelmointikieliä. Se perustuu C-kielelle, johon on tehty laajennuksia kuten olio-ohjelmointiin liittyvät mekanismit. C++ -kielen kehitys alkoi 1980-luvulla Bjarne Stroustrupin johdolla työnimellä C with Classes. Kieltä kehitettiin vähitellen lisäämällä siihen ominaisuuksia ja epävirallisia standardeja. Ensimmäisen virallisen virallisen standardin C++ saavutti vuonna 1997. (7, s. 16 - 17)

Olio-ohjelmoinnin yhteydessä puhutaan olioista ja luokista. Luokka sisältää koodin, joka vastaa tiettyä kokonaisuutta tai tehtävää. Esimerkiksi tämän työn yhteydessä tehdyssä Mittaus-sovelluksessa on SqlRead-luokka, joka hoitaa tietojen lukemisen Mittaus-tietokannasta. Olio on luokan ilmentymä, jonka kautta luokan ominaisuuksia käytetään. Luokista voi rakentaa luokkakirjastoja, jolloin valmiina olevia ominaisuuksia on helppo käyttää uudelleen tarpeen mukaan. (8.)

5.4 Qt4

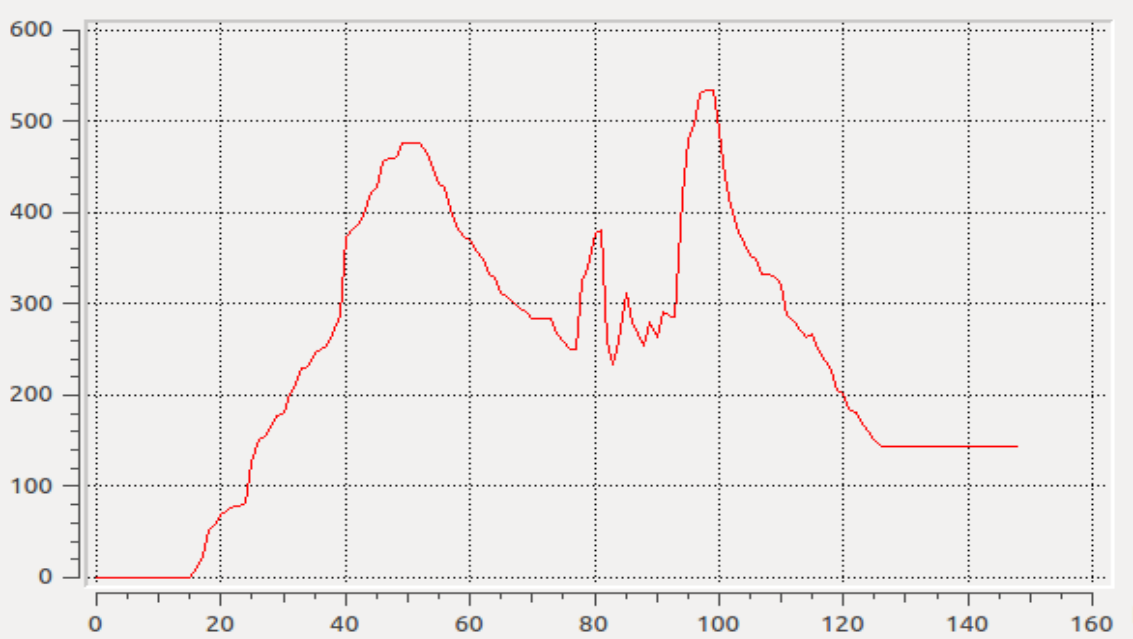
Qt4 on kehitysympäristö, joka toimii yleisimmillä käyttöjärjestelmillä. Se sisältää sisäänrakennetun C++ -luokkakirjaston eli sitä käytetään pääasiallisesti C++ -kielisen koodin tuottamiseen. Qt4:stä löytyy kuitenkin myös tuki joillekin muille kielille. Qt4 oli alun perin Nokian Symbian-käyttöjärjestelmän kehitystyökalu. Symbianin kehitys kuitenkin lopetettiin ja Digia osti Qt4:n kokonaisuudessaan Nokialta vuonna 2011. (9.)

Qt4 sisältää Qt Creator -ohjelimointiympäristön, jonka Designer ominaisuutta käyttäen on helppo luoda visuaalisia käyttöliittymiä ja määrittää niiden ominaisuuksia Qt Designer sisältää paljon valmiita komponentteja (widgets), kuten painikkeita, tekstikenttiä ja valikoita, joita hyödynnettiin Mittaus-sovelluksen kehittämisessä. (9.)

5.5 Qwt

Qwt on luokkakirjasto, jonka saa lisättyä Qt Designeriin. Se sisältää erilaisia komponentteja, joita voi käyttää apuna visuaalisen käyttöliittymän luonnissa. Qwt:n saa ladattua ilmaiseksi osoitteesta http://sourceforge.net/projects/qwt tai Linux-käyttöjärjestelmissä Synaptic paketinhallintaohjelmistolla. (10.)

Mittaus-sovelluksessa käytetään Qwt:n käyränpiirto-ominaisuutta eli plotteria (kuva 9), koska Qt Creatorista ei löydy valmiina vastaavanlaista. Kun Qwt:n asentaa Qt Creatorin yhteyteen, antaa se mahdollisuuden käyttää siinä olevia valmiita luokkia ja käyttöliittymän komponentteja.



Kuva 9. QwtPlot-koordinaatisto

Kuvan 9 käyrän piirtoon ja koordinaatiston luontiin tarvitaan Qwt-kirjaston luokkia QwtPlotCurve, QwtPlotGrid ja QwtPlot.

5.5.1 QwtPlotCurve

QwtPlotCurve-luokka määrittää piirrettävien käyrien tyypin ja värin. Kuvan 9 käyrän tiedot määritellään esimerkkikoodin 5 mukaisesti.

```
Curve = new QwtPlotCurve();
Curve->setPen(QPen(Qt::red));
Curve->attach(ui->qwtPlot);
```

Esimerkkikoodi 5

Ensin luokasta QwtPlotCurve luodaan Curve-niminen olio. Seuraavaksi sen väriksi määritetään punainen ja lopuksi se kiinnitetään käyttöliittymän qwtPlot-koordinaatistoon.

5.5.2 QwtPlotGrid

QwtPlotGrid-luokan avulla qwtPlot-koordinaatistoon luodaan ruudukko. Ruudukolle voi määrittää esimerkiksi värin, viivan tyypin, käytetyt akselit ja ruudukon koon. Kuvan 9 ruudukon määritys tapahtuu esimerkkikoodin 6 komennoilla:

```
Grid = new QwtPlotGrid();
Grid->setPen(QPen(Qt::black, 0, Qt::DotLine));
Grid->enableX(true);
Grid->enableY(true);
Grid->attach(ui->qwtPlot);
```

Esimerkkikoodi 6

Ensin luokasta QwtPlotGrid luodaan Grid-niminen olio. Ruudukon tyyliksi määritetään musta pisteviiva. Seuraavaksi ruudukon x- ja y-akselit otetaan käyttöön. Lopuksi ruudukko kiinnitetään käyttöliittymän qwtPlot-koordinaatistoon.

5.5.3 QwtPlot

QwtPlot-luokka sisältää käyttöliittymäkomponentin, jonka avulla luodaan koordinaatisto, johon käyrät piirretään. Esimerkkikoodissa 7 suoritetaan käyrän piirto koorrdinaatistoon.

```
void MainWindow::PiirraPlotteriin()
{
    x_akseli[indeksi]=indeksi;
    y_akseli[indeksi]=arvoY;
    Curve->setSamples(x_akseli, y_akseli, indeksi);
    ui-> qwtPlot->replot();
}
```

Esimerkkikoodi 7

Komento "`ui-> qwtPlot->replot();`" antaa käskyn piirtää käyrän `Curve` x- ja y-akselien arvot qwtPlot-koordinaatistoon. Muuttuja `indeksi` on kulloisenkin tarkstelupisteen järjestysnumero, jonka perusteella käyrä piirretään. Muuttujat `x_akseli` ja `y_akseli` saavat jokaisessa tarkastelupisteessä uudet arvot, jotka piirretään plotteriin. Tässä tapauksessa `x_akseli`:lla on sama arvo kuin tarkastelupisteen järjestysnumero tällöin `y_akseli` saa kulloisessakin tarkastelupisteessä arvokseen luvun `arvoY`.

5.6 Doxygen

Doxygen on koodin dokumentointiin käytetty ilmainen työkalu. Se on kehitetty C++ koodin dokumentointia varten mutta sillä voi dokumentoida myös monia muita tunnettuja ohjelmointikieliä. Doxygenia on kehittänyt Dimitri van Heesch vuodesta 1997 alkaen. (11)

Mittaus-sovelluksen koodi on dokumentoitu Doxygenilla. Tuotetun koodin huolellinen dokumentointi on hyvän ohjelmointitavan mukaista ja myös tärkeää koodin ymmärrettävyyden ja käytettävyyden kannalta. Dokumentoinnista on pyritty tekemään

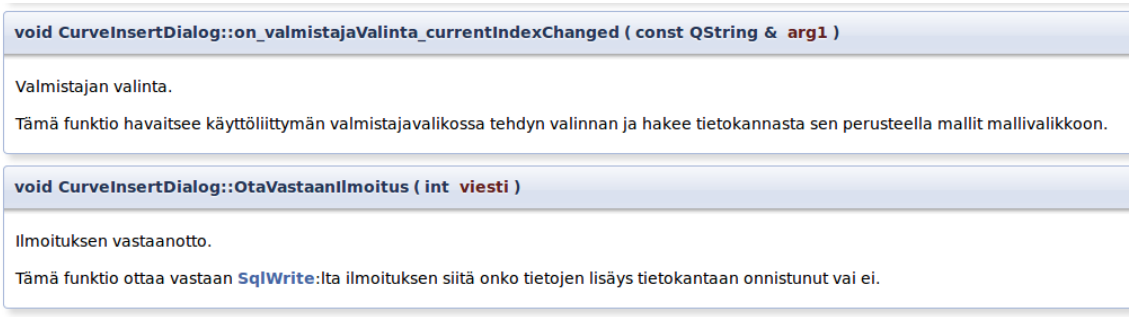
mahdollisimman kattava, jotta sovelluksen kehittämistä myöhemmin jatkava taho ymmärtäisi mahdollisimman helposti eri toiminnallisuudet.

Doxygen huomaa automaattisesti sellaisen koodin, johon on laitettu dokumentoinnin mahdollistavat tunnisteet. Kuvassa 10 tunnisteet ja niiden sisältämä teksti näkyvät sinisellä.

```
30  //! Valmistajan valinta.
31  /*!
32  Tämä funktio havaitsee käyttöliittymän valmistajavalikossa tehdyn valinnan ja hakee
33  tietokannasta sen perusteella mallit mallivalikkoon.
34  */
35  void CurveInsertDialog::on_valmistajaValinta_currentIndexChanged(const QString &arg1)
36  {
37      valmistaja = arg1;
38      if(ui->valmistajaValinta->currentIndex()>>0){
39          emit ValmistajaValinta(valmistaja);
40      }
41  }
42  //! Mallin valinta.
43  /*!
44  Tämä funktio havaitsee käyttöliittymän mallivalikossa tehdyn valinnan ja tallettaa sen
45  perusteella myöhemmin syötetyt esikäyrän arvot.
46  */
47  void CurveInsertDialog::on_malliValinta_currentIndexChanged(const QString &arg1)
48  {
49      malli = arg1;
50      emit MalliValinta(malli);
51      if(ui->malliValinta->currentIndex()>>0){
52          emit MalliValinta(malli);
53      }
54  }
```

Kuva 10. Doxygen tunnisteet

Doxygen luo dokumentoidusta koodista HTML-tiedoston, jossa tunnisteiden sisällä oleva teksti on kyseisten muuttujien, funktioiden tai luokkien tietokentässä (Kuva 11). HTML-tiedoston avulla dokumentoitua koodia on selkeä tarkastella kun koko projektin kaikki osat löytyvät samasta paikasta hyvässä järjestyksessä.



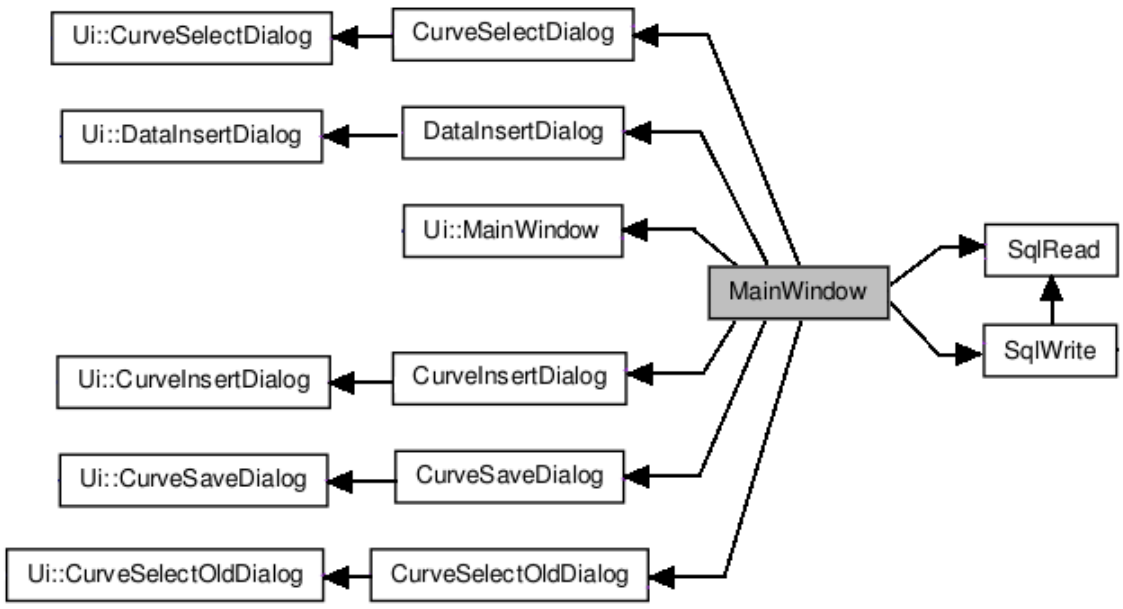
Kuva 11. Funktioiden tietokentät HTML-dokumentoinnissa

6 Mittaus-sovellus

Mittaus-sovellus on tätä työtä varten kehitetty ohjelma, jonka avulla Mittaus-tietokannan tietoja voi käsitellä. Ohjelma kehitettiin Veranos Oy:n antamien toiveiden mukaisesti siten, että kaikki suunnitellut ominaisuudet pyrittiin sisällyttämään siihen mahdollisimman toimivasti.

Ohjelma koostuu kahdeksasta luokasta, joista kaksi käsittelee Mittaus-tietokantaan kirjoittamisen ja sieltä lukemisen. Loput kuusi luokkaa muodostavat käyttöliittymän, joka esittää vanhat mittauskäyrät, esikäyrät ja mittausten simuloinnin sekä hoitaa Mittaus-tietokannan käsittelyn käyttäjän käskyjen mukaan.

Kuvassa 12 on esitettynä Mittaussovelluksen luokkakaavio, josta voi havaita miten ohjelmisto rakentuu. MainWindow ja sen vasemmalla puolella olevat luokat muodostavat käyttöliittymän. Ui:: ja luokan nimi viittaa siihen, että kyseinen luokka tuottaa osan käyttöliittymästä (Ui = user interface eli käyttöliittymä). MainWindown oikealla puolella olevat luokat hoitavat yhteydet Mittaus-tietokannan kanssa.



Kuva 12. Mittaus-sovelluksen luokkakaavio

6.1 Käyttöliittymä

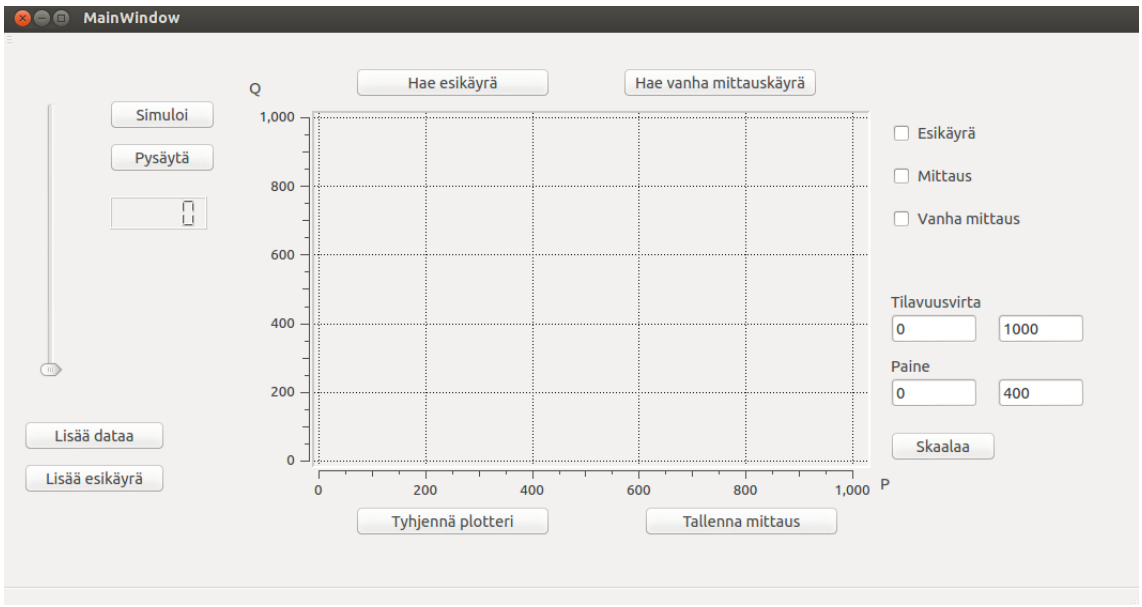
Mittaus-sovelluksen käyttöliittymä koostuu perusnäkymästä ja viidestä avattavasta valikosta. Valikot avautuvat perusnäkymässä olevilla painikkeilla, joiden avulla hoidetaan esimerkiksi vanhojen mittauskäyrien ja esikäyrien hakeminen sekä tietojen lisääminen Mittaus-tietokantaan. Perusnäkymässä olevassa koordinaatistossa esitetään mittausten simulointi, esikäyrät ja vanhat mittauskäyrät.

Monissa käyttöliittymän ikkunoissa on valikoita, joiden avulla Mittaus-tietokannassa olevaa tietoa käsitellään. Valikot toimivat siten, että ensin valitaan minkä valmistajan pumpusta on kyse. Valinnan perusteella ohjelma hakee tietokannasta kyseisen pumpun kaikki mallit ja lisää ne mallivalikkoon. Kun jokin malli valitaan, hakee ohjelma taas kyseisen mallin sarjanumerot tai esikäyrät ja niin edelleen.

6.1.1 MainWindow

Käyttöliittymän perusnäkymä on MainWindow-luokan toteuttama ikkuna (Kuva 13). Tässä ikkunassa on mahdollisuus simuloida mittauksia ikkunan vasemmassa reunassa olevan palkin sekä Simuloi- ja Pysäytä-nappien avulla. Simuloitu mittaus piirtyy käyränä ikkunan keskellä olevaan koordinaatistoon. Koordinaatistoon piirretään myös esikäyrät ja vanhat mittauskäyrät, jotka voi hakea tietokannasta perusnäkymän painikkeilla aukeavista valikoista. Ikkunan oikeassa yläkulmassa olevista valintapainikkeista voi määrittää, mitkä valituista käyristä näytetään koordinaatistossa.

Koordinaatiston esitysalueen voi skaalata haluamakseen, syöttämällä oikeassa reunassa oleviin kenttiin haluamansa x- ja y-akselien arvot ja painamalla Skaalaa-nappia. Tyhjennä plotteri -nappia painamalla koordinaatisto tyhjennetään ja skaalataan alkuarvojen mukaiseksi.



Kuva 13. MainWindow

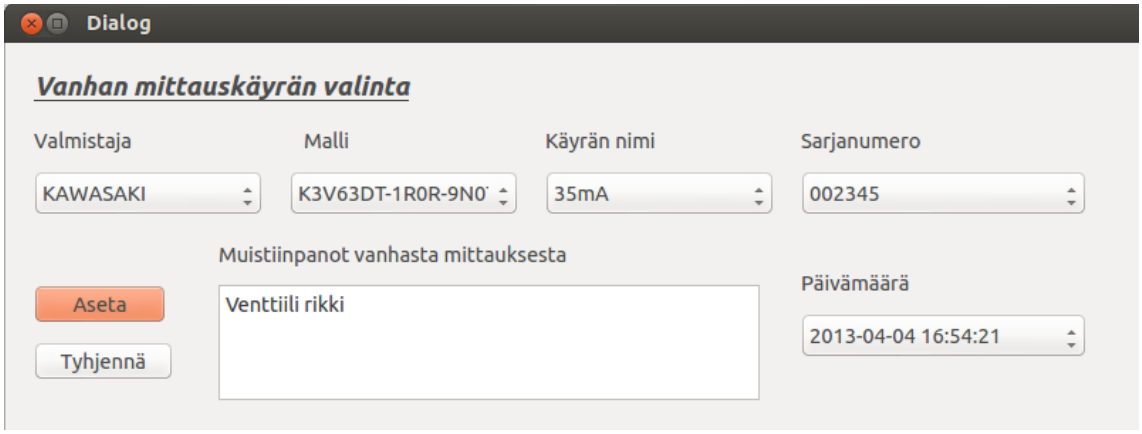
6.1.2 CurveSelectDialog

CurveSelectDialog-luokka toteuttaa *Esikäyrän valinta* -ikkunan (kuva 14). Ikkuna avautuu MainWindow:n Hae esikäyrä -painikkeella. Tässä ikkunassa määritetään esikäyrä, jonka perusteella mittaus suoritetaan. Ikkunan valikoista valitaan esikäyrän tiedot eli pumpun valmistaja, pumpun malli ja esikäyrän nimi. Aseta-nappia painamalla ohjelma hakee esikäyrän arvot tietokannasta ja valittu esikäyrä piirtyy MainWindow:n koordinaatistoon.

Kuva 14. Esikäyrän valinta -ikkuna

6.1.3 CurveSelectOldDialog

CurveSelectOldDialog-luokka toteuttaa *Vanhan mittauskäyrän valinta* -ikkunan (kuva 15), jonka perusteella tietokannasta haetaan aiemmin suoritettujen mittausten käyriä. Ikkuna avautuu MainWindow:n Hae vanha mittauskäyrä -painikkeella. Vanhaa mittauskäyrää valittaessa on määriteltävä pumpun valmistaja, malli, mitä esikäyrää mittaus kuvaa, pumpun sarjanumero, jolla mittaus on suoritettu, sekä mittauksen ajankohta. Muistiinpanokentään tulee tieto mittauksen yhteydessä kirjoitetuista muistiinpanoista. Aseta-nappia painamalla vanha mittauskäyrä piirtyy MainWindow:n koordinaatistoon. Tyhennä-nappi tyhjentää valikot, jolloin valitsemisen voi aloittaa alusta.



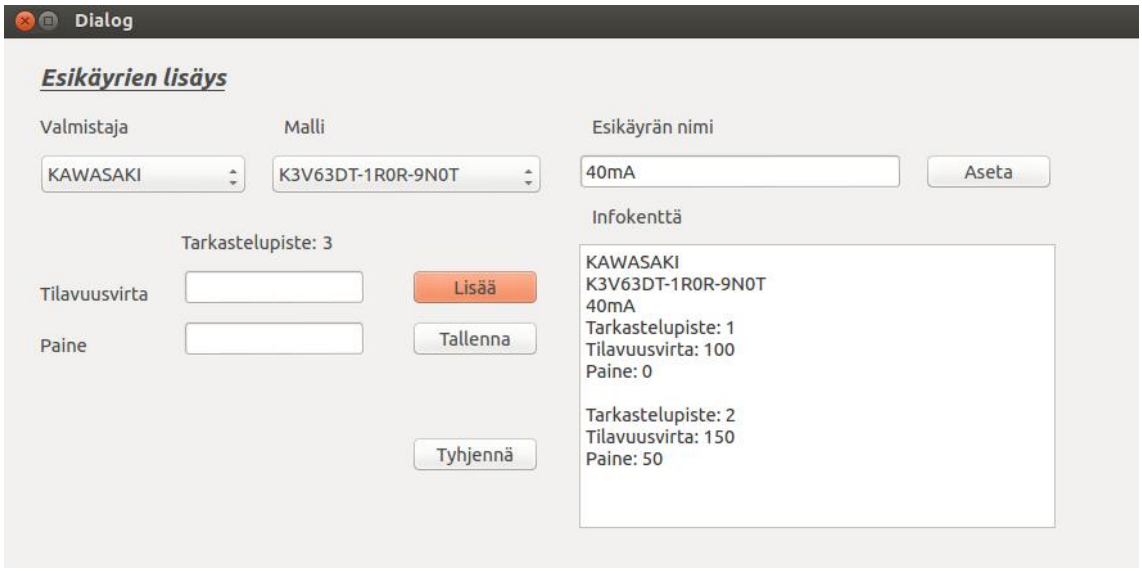
Kuva 15. Vanhan mittauskäyrän valinta -ikkuna

6.1.4 CurveInsertDialog

CurveInsertDialog-luokka toteuttaa *Esikäyrien lisäys* -ikkunan (kuva 16) joka, aukeaa MainWindow:n Lisää Esikäyrä -painikkeella. Tässä ikkunassa tietokantaan voi syöttää uusia esikäyriä. Ensin valikoista on valittava pumpun valmistaja ja malli, jolle esikäyrä syötetään ja annettava käyrälle nimi. Aseta-painikkeella tiedot hyväksytään ja esikäyrän arvojen syöttäminen mahdollistuu.

Esikäyrän arvot lisätään tarkastelupiste kerrallaan siten, että kyseisen pisteen paine ja tilavuusvirta syötetään niille varattuihin kenttiin ja painetaan Lisää-nappia. Näin tarkastelupisteelle annetut arvot siirtyvät näkymään Infokentässä ja uusien arvojen syöttäminen on mahdollista. Kun käyrän kaikki arvot on annettu, ne voi tallentaa tietokantaan Tallenna-painikkeella. Kun tallennus on suoritettu, tietokanta lähettää vielä

varmistuksen onko tiedot varmasti saatu tallennettua. Ilmoitus tulee näkyviin Infokenttään. Jos tietoja ei halua tallentaa esimerkiksi syöttövirheen takia, voi tietojen syöttämisen aloittaa alusta Tyhjennä-nappia painamalla.



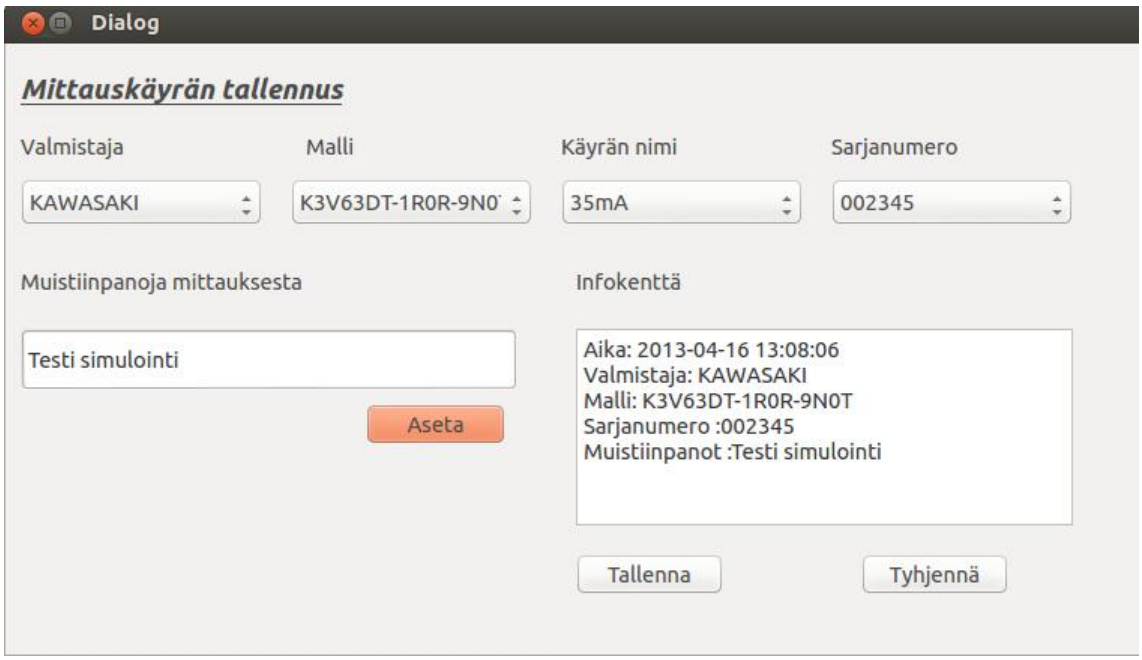
Kuva 16. CurveInsertDialog-ikkuna

6.1.5 CurveSaveDialog

CurveSaveDialog-luokka toteuttaa *Mittauskäyrän tallennus* -ikkunan (kuva 17). Se avautuu MainWindow:n Tallenna mittaus -painikkeella. Tässä ikkunassa tallennetaan pumpulle tehty mittaus, joka on piirretty MainWindown koordinaatistoon. Tallennusta varten tarvitaan tiedot pumpun valmistajasta, mallista, sarjanumerosta ja siitä, mitä esikäyrää mittaus kuvastaa. Tallennuksen yhteydessä on myös mahdollista lisätä mittaukseen liittyviä muistiinpanoja.

Kun tarvittavat tiedot on syötetty valintakenttiin, ne voi hyväksyä Aseta-nappia painamalla. Tällöin tiedot siirtyvät Infokenttään tarkasteltaviksi. Ohjelma lisää päivämäärän ja ajan automaattisesti tietoihin, koska myös sitä tarvitaan mittauskäyrän tallentamiseen.

Kun mittauskäyrän kaikki tiedot on annettu, ne voi tallentaa tietokantaan Tallenna-painikkeella. Kun tallennus on suoritettu, tietokanta lähettää vielä varmistuksen, että tiedot varmasti saatu tallennettua.

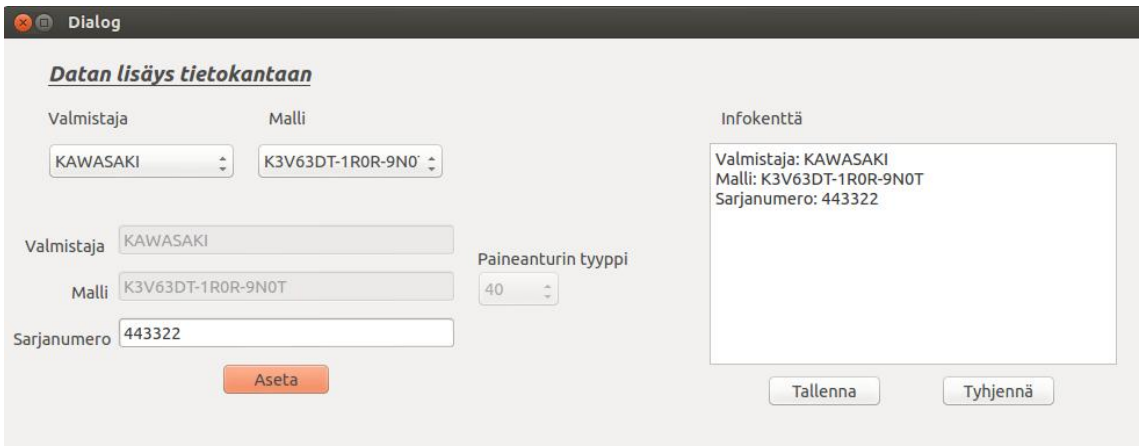


Kuva 17. Mittauskäyrän tallennus -ikkuna

6.1.6 DataInsertDialog

DataInsertDialog-luokka toteuttaa *Datan lisäys tietokantaan* -ikkunan (kuva 18), joka avautuu MainWindown Lisää dataa -painikkeella. Tässä ikkunassa tietokantaa voi syöttää uusia tietoja pumpun valmistajasta, mallista, sarjanumerosta ja siitä, millä paineanturityypillä mittaukset kyseiselle pumpulle suoritetaan.

Syötetty tieto hyväksytään painamalla Aseta-nappia, jolloin infokenttään tulee näkymään, mitä ollaan tallentamassa. Tallennus suoritetaan Tallenna-nappia painamalla. Kun tallennus on suoritettu, lähettää tietokanta vielä varmistuksen, että tiedot on varmasti saatu tallennettua.



Kuva 18. Datan lisäys tietokantaan -ikkuna

6.2 Tietokannan käsittely

Mittaus-tietokannassa olevien tietojen käsittely hoidetaan SqlRead- ja SqlWrite-luokkien avulla. Nämä luokat sisältävät Sql-kieliset komennot, joiden avulla ne muokkaavat Mittaus-tietokannan sisältöä. Ohjeet toimia saadaan käyttöliittymältä tulevien valintojen ja syötteiden perusteella.

6.2.1 SqlRead

SqlRead-luokka muodostaa yhteyden Mittaustietokantaan AvaaDatabase funktiolla (esimerkkikoodi 8). Yhteyden muodostaminen vaatii toimiakseen PSQL ajurit, jotka asennetaan Qt:n yhteyteen.

```
void SqlRead::AvaaDatabase()
{
    if(!db.open()){
    db = QSqlDatabase::addDatabase("QPSQL");
    db.setUserName("postgres");
    db.setHostName("localhost");
    db.setPort(5432);
    db.setPassword("12345678");
    db.setDatabaseName("Mittaus");
}
```

Esimerkkikoodi 8

Funktiossa annetaan tarvittavat tiedot yhteyden muodostamiselle kuten tietokannan nimi ja salasana.

Kun yhteys on muodostettu, voi Mittaus-tietokannassa olevia tietoja käsitellä. SqlRead-luokka keskittyy tietojen lukemiseen. Kun käyttöliittymässä tehdään valintoja, jotka edellyttävät tiedon hakemista tietokannasta, tämä luokka hoitaa prosessin. Esimerkiksi jos jossain käyttöliittymän valikossa valitaan pumpun valmistajaksi Kawasaki, tulee tieto valinnasta tälle luokalle. Valmistajan nimen perusteella tämä luokka osaa hakea Mittaus-tietokannasta kaikki mallit, joita Kawasakilta on tallennettu, ja lähettää ne käyttöliittymän mallinvalintakenttään. Myös esikäyrien ja vanhojen mittauskäyrien hakeminen hoidetaan tämän luokan avulla.

6.2.2 SqlWrite

SqlWrite-luokka hoitaa uuden tiedon kirjoittamisen Mittaus-tietokantaan. Kun tietokantaan halutaan lisätä tietoa, kuten uusia valmistajia tai malleja, se tapahtuu tämän luokan toimesta. Myös mittauskäyrien tallentaminen ja esikäyrien syöttö tapahtuu tämän luokan välityksellä. Käyttöliittymältä tulevat syötteet otetaan vastaan ja kirjoitetaan Mittaus-tietokantaa Sql-kielisillä komennoilla. Esimerkiksi uuden valmistaja lisääminen tapahtuu esimerkkikoodin 9 esittämällä tavalla.

```
void SqlWrite::LisaaValmistaja(QString uusiValmistaja)
{
        QSqlQuery querylite(db);
        querylite.prepare("INSERT INTO make(id, name) VALUES (DEFAULT,
?)");
        querylite.addBindValue(uusiValmistaja);
        if(querylite.exec()==true){
        viesti=1;
        }
        else{viesti=0;}
        emit ilmoitus(viesti);
}
```

Esimerkkikoodi 9

Koodissa funktio LisaaValmistaja(QString uusiValmistaja) saa tiedon uuden valmistajan nimestä muuttujassa QString uusiValmistaja. Tieto kirjoitetaan Mittaus-tietokannan make-tauluun komennolla `”INSERT INTO make (id, name) VALUES (DEFAULT, ?)”`. Valmistajan nimi lisätään kysymysmerkin paikalle komennolla `querylite.addBindValue(uusiValmistaja)`. Jos tieto tallentuu tietokantaan onnistuneesti, lähetetään käyttöliittymälle ilmoitus int viesti -muuttujassa. Muuttuja voi

saada arvoksi 1, joka tarkoittaa, että tallennus on onnistunut tai 0, jolloin tallennus on epäonnistunut. Käyttöliittymän luokat osaavat tulkita nämä arvot ja kirjoittaa Infokenttiin tiedon tallennuksen tuloksesta.

7 Yhteenveto

Tietokantarakenteet eivät olleet tämän insinöörityön tekijälle ennestään tuttuja, mikä loi haasteita projektin aikana. Huolellisen aheeseen perehtymisen lopputuloksena saatiin kuitenkin aikaan toimiva sovellus, jonka avulla tietokannassa olevia tietoja voi muokata tarpeen mukaan.

Sovelluksessa oleva mittausten ja esikäyrien piirto-ominaisuus oli kehitetty jo tätä työtä alustavan projektin yhteydessä. Tämän työn haasteena oli saada tietokantaan tallennettujen käyrien arvot luettua niin, että ne saadaan piirrettyä oikein koordinaatistoon. Myös mahdollisuus tallentaa tehdyt mittaussimuloinnit sellaiseen muotoon, että ne ovat myöhemmin luettavissa, vaati runsaasti työtä.

Vaikka sovellus toimiikin, ei se sellaisenaan riitä parantamaan Veranos Oy:n huoltopalvelujen laatua. Lopulliset mittausjärjestelyt, joihin ei tässä työssä otettu kantaa, on toteutettava huolellisesti erillisenä projektina ja liitettävä osaksi nykyistä sovellusta, jotta todellinen hyöty saavutettaisiin. Tämä työ on pyritty dokumentoimaan niin kattavasti, että työn jatkajan olisi mahdollisimman helppo ymmärtää nykyisen ohjelman rakenteita ja toimintaa. Toimivat mittausjärjestelyt yhdistettynä tähän työhön, voivat hyvinkin tuoda uudenlaista tehokuutta ja kilpailuetua Veranos Oy:n huoltopalveluihin.

Lähteet

1 Hovi, Ari; Huotari, Jouni; Tapio, Lahdenmaäki. 2005. Tietokantojen suunnittelu ja indeksointi, Docendo, Porvoo.

2 Douglas, Korry; Douglas, Susan. 2003. PostgreSQL, Sams Publishing, Yhdysvallat.

3 w3schools.com, ohjelmoinnin opetussivusto, <http://www.w3schools.com/sql/sql_select.asp> Viitattu 16.4.2013.

4 w3schools.com, ohjelmoinnin opetussivusto, <http://www.w3schools.com/sql/sql_where.asp> Viitattu 16.4.2013.

5 Ubuntun kotisivut, <http://www.ubuntu.com> Viitattu 18.4.2013

6 Synapticin kotisivut, <http://www.nongnu.org/synaptic/> Viitattu 18.4.2013.

7 Hietanen, Päivi. 2008. C++ ja olio-ohjelmointi, WSOY, Vantaa.

8 Rintala, Matti; Jokinen, Jyke. 2005. Olioiden ohjelmointi C++:lla, Satku, Jyväskylä.

9 Qt:n kotisivut <http://qt.digia.com/> Viitattu 20.4.2013.

10 Qwt:n kotisivut <http://qwt.sourceforge.net/> Viitattu 20.4.2013.

11 Doxygenin kotisivut <http://www.stack.nl/~dimitri/doxygen/index.html> Viitattu 20.4.2013